week6_eda2020

Lucia M Rodriguez Bravo

March, 2020

Contents

Graphing Data activity

Data extraction

First we're calling the packages to be used

```
library(tidyverse)
library(dplyr)
library(lubridate)
```

Load data

Second, we're loading a database already included in R

```
irisdata<-iris #I chose to create an object to look at the data more easily
# ----------------------------------------
#STEP 1: Did it load correctly?
head(irisdata) #loading correct
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
```

```
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# ----------------------------------------
#STEP 2: Are the data types right?
sapply(irisdata,class) #classes correct
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##    "numeric"    "numeric"    "numeric"    "numeric"     "factor"
```

```
# ----------------------------------------
# STEP 3: Check for missing or impossible numeric values
range(irisdata$Sepal.Length)
```

```
## [1] 4.3 7.9
```

```
range(irisdata$Sepal.Width)
```

```
## [1] 2.0 4.4
```

```
range(irisdata$Petal.Length)
```

```
## [1] 1.0 6.9
```

```
range(irisdata$Petal.Width) #all good
```

```
## [1] 0.1 2.5
```

```
# ----------------------------------------
# STEP 4: Check factor levels
levels(irisdata$Species)
```

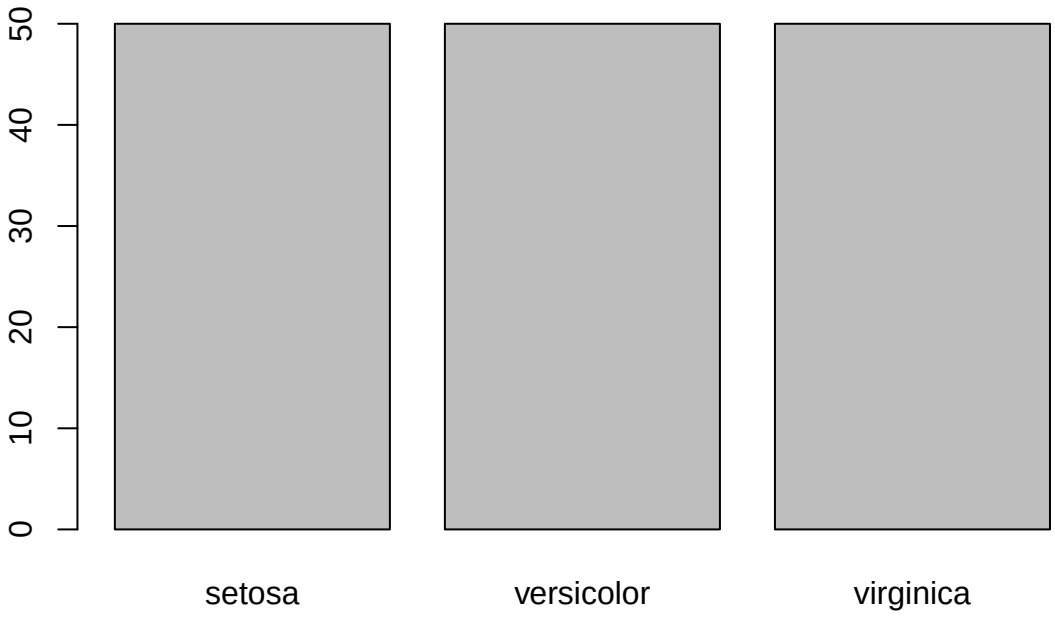
```
## [1] "setosa"     "versicolor" "virginica"
```

```
# All OK
```

Plot One Variable - Base

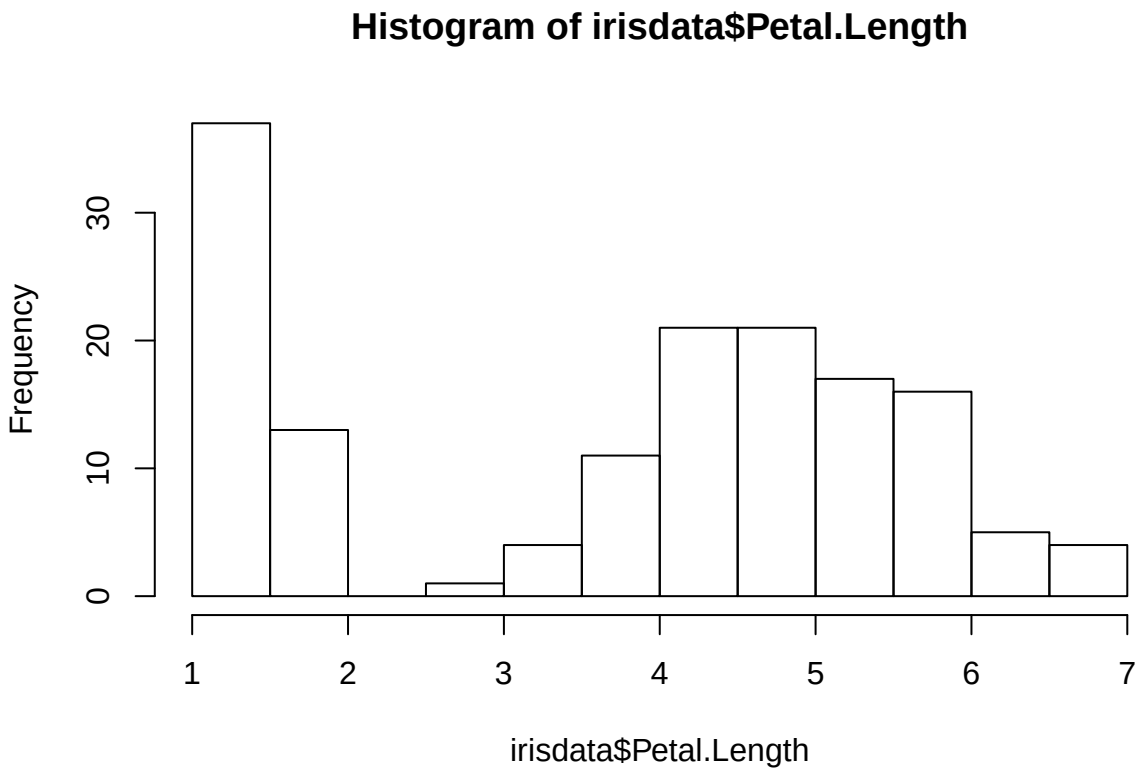
Discrete variable

```
#To plot counts (Y) of a discrete variable X (in this case species of flower) in Base
barplot(table(irisdata$Species)) #table creates a count
```



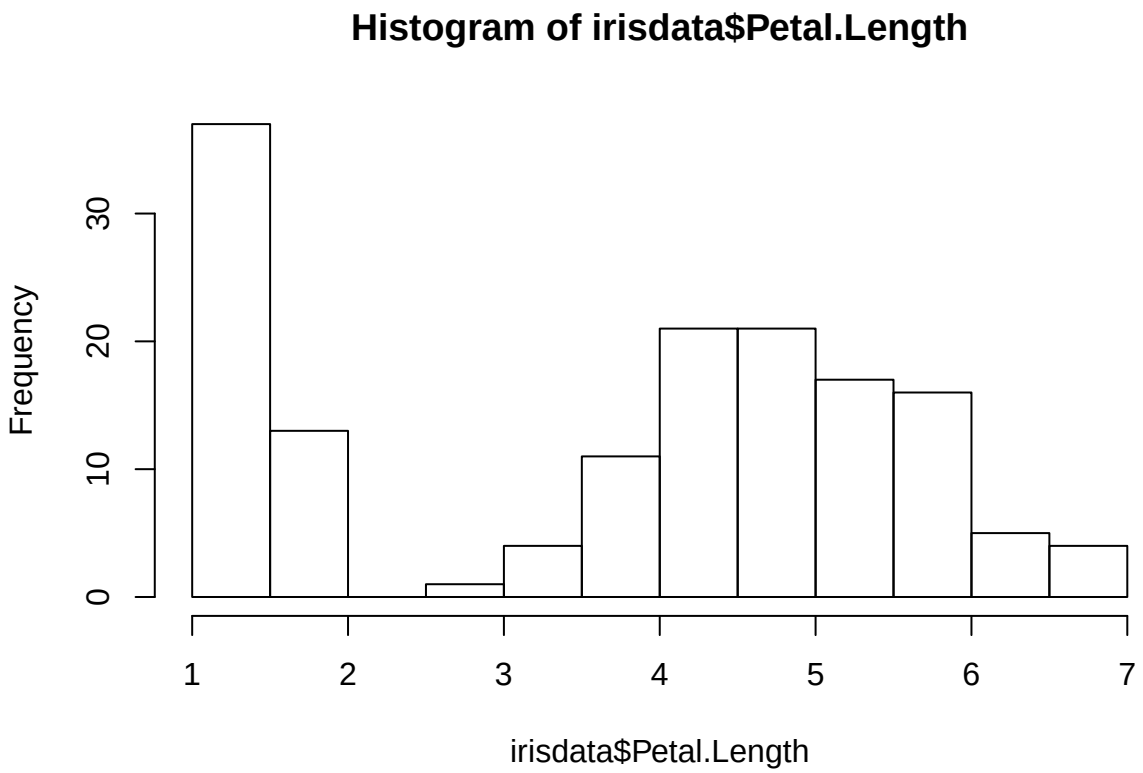
Continuous variable

```
#To plot a continuous variable X (in this case petal length) in Base
hist(irisdata$Petal.Length)
```




```
# Try choosing different breaks or bins of data
hist(irisdata$Petal.Length,breaks=4)
```

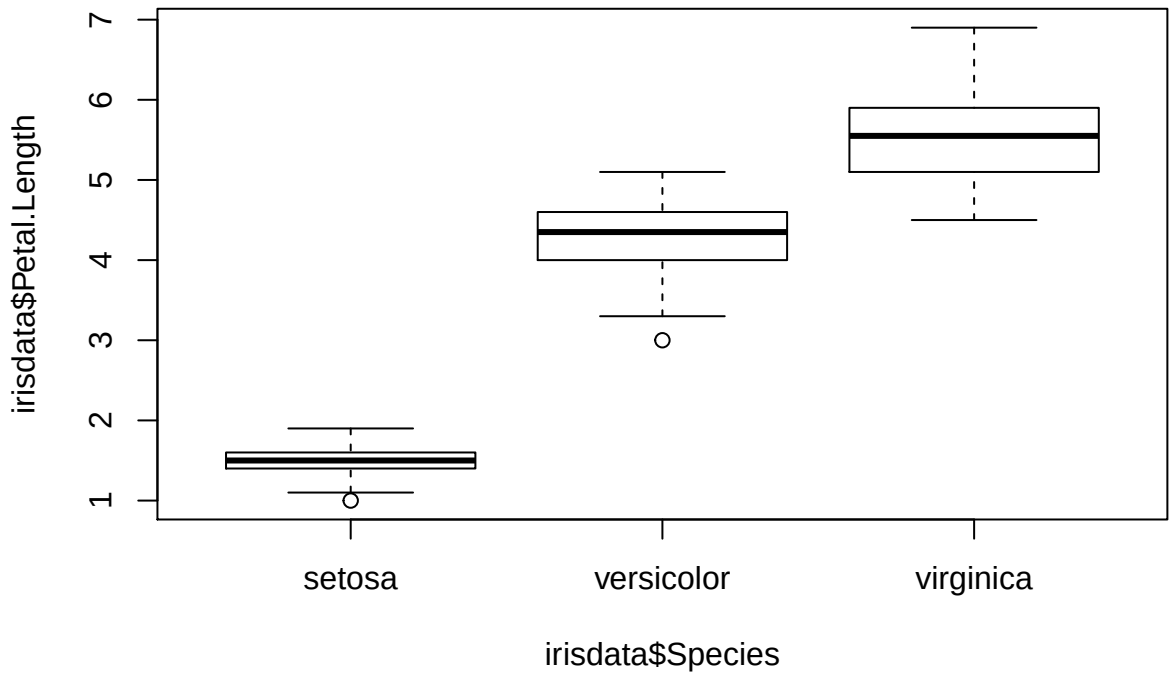

```
#A histogram with four columns
# Extract the hist function calculations
petalhist<-hist(irisdata$Petal.Length) #we create an object in which we can see our histogram function
```



```
petalhist
```

```
## $breaks
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
##
## $counts
##  [1] 37 13  0  1  4 11 21 21 17 16  5  4
##
## $density
##  [1] 0.49333333 0.17333333 0.00000000 0.01333333 0.05333333 0.14666667
##  [7] 0.28000000 0.28000000 0.22666667 0.21333333 0.06666667 0.05333333
##
## $mids
##  [1] 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75 5.25 5.75 6.25 6.75
##
## $xname
## [1] "irisdata$Petal.Length"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```
# A way to represent by species more clearly
boxplot(irisdata$Petal.Length~irisdata$Species)
```

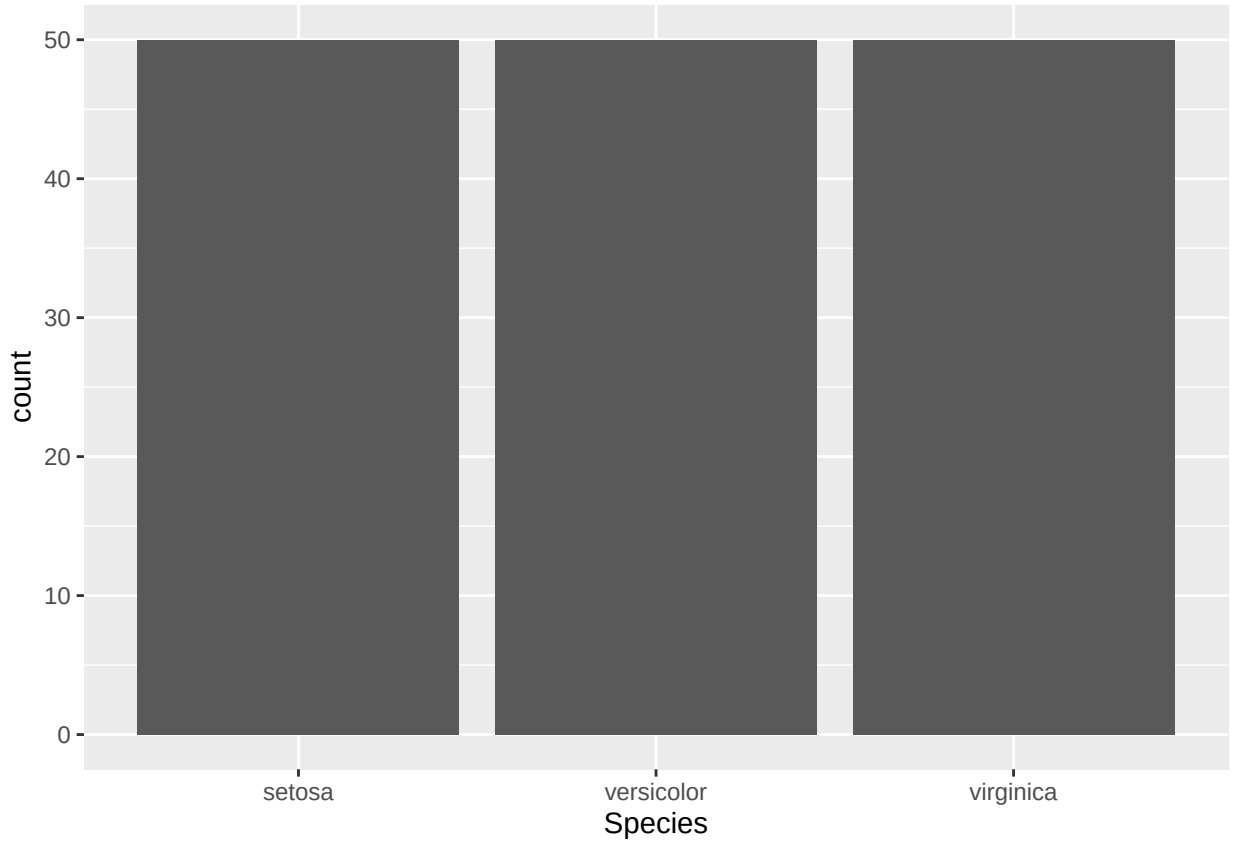


Plot One Variable - ggplot

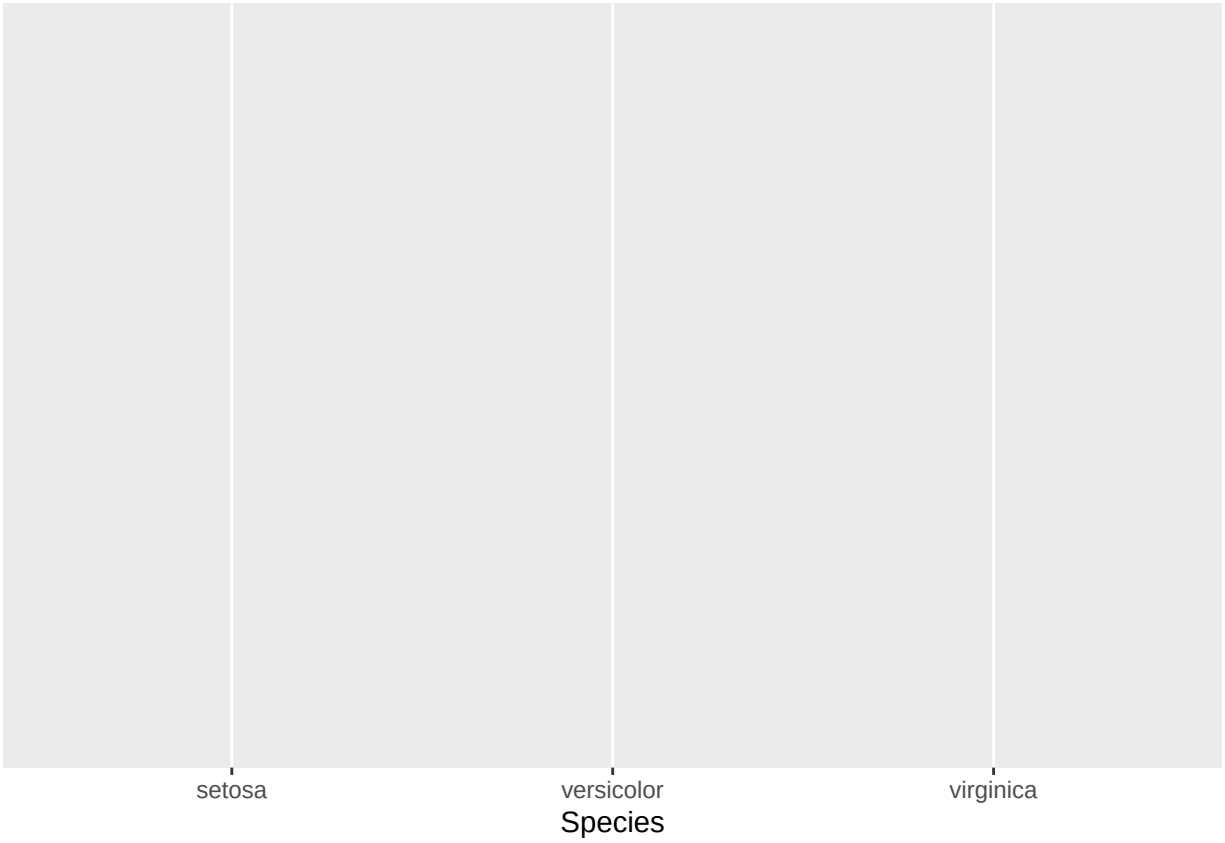
Discrete variable

```
#To plot counts (Y) of a discrete variable X (in this case species of flower) in ggplot

Speciesgg<-ggplot(data = irisdata,aes(Species))+geom_bar()
Speciesgg #I chose to store the plot as an object
```

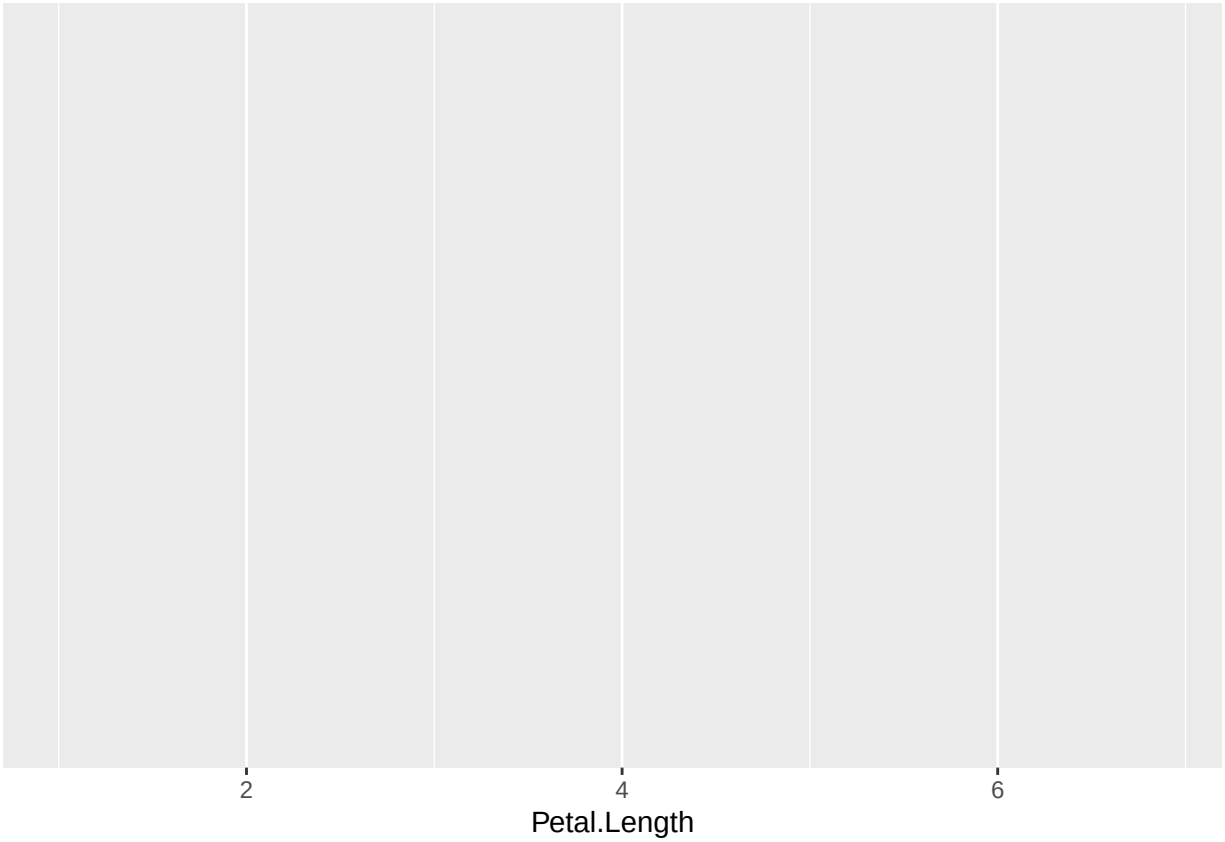



```
# Demonstrate how things can be layered (by creating a plot object and adding different types of plots
Speciesbase<-ggplot(data = irisdata,aes(Species))
Speciesbase #This is just the canvas
```



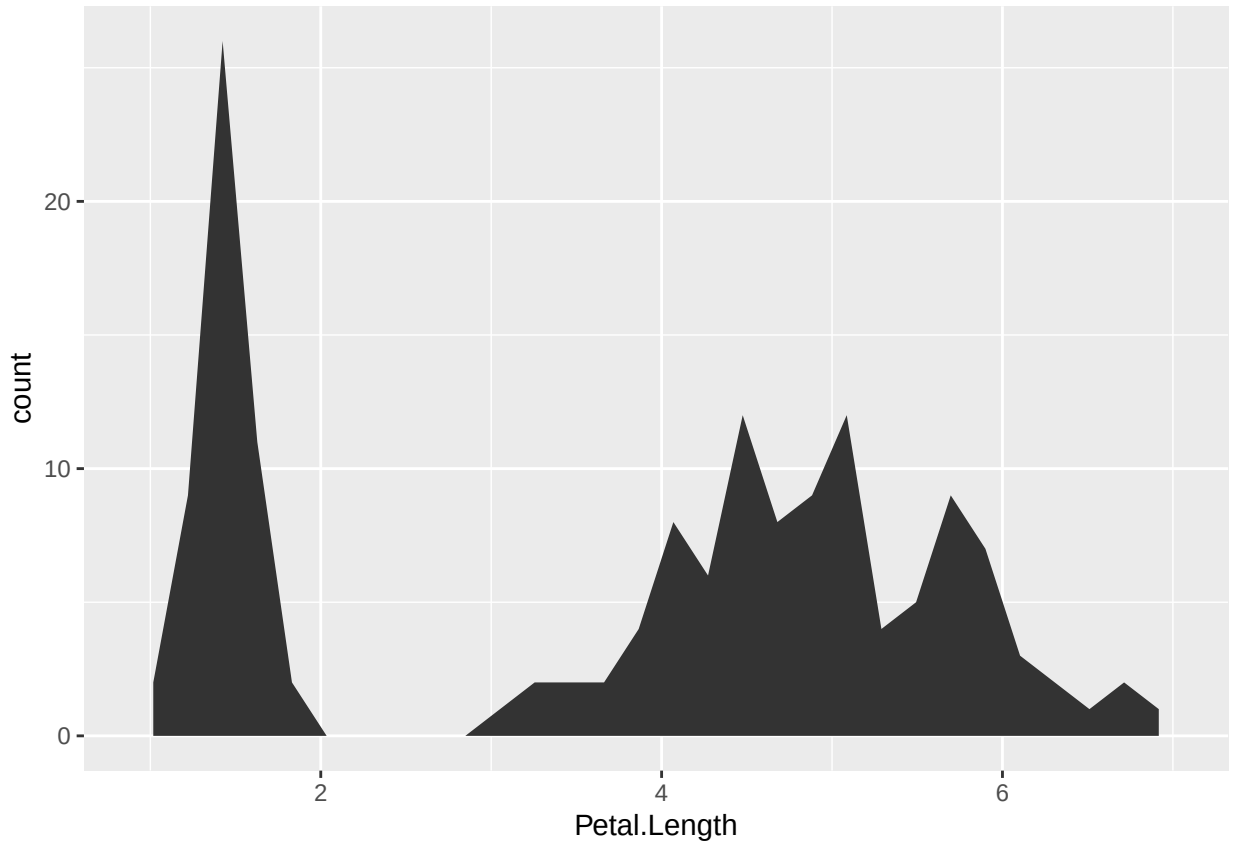
Continuous variable

```
#To plot a continuous variable X (in this case petal length) in ggplot
petalgg<-ggplot(data=irisdata,aes(Petal.Length))
petalgg
```

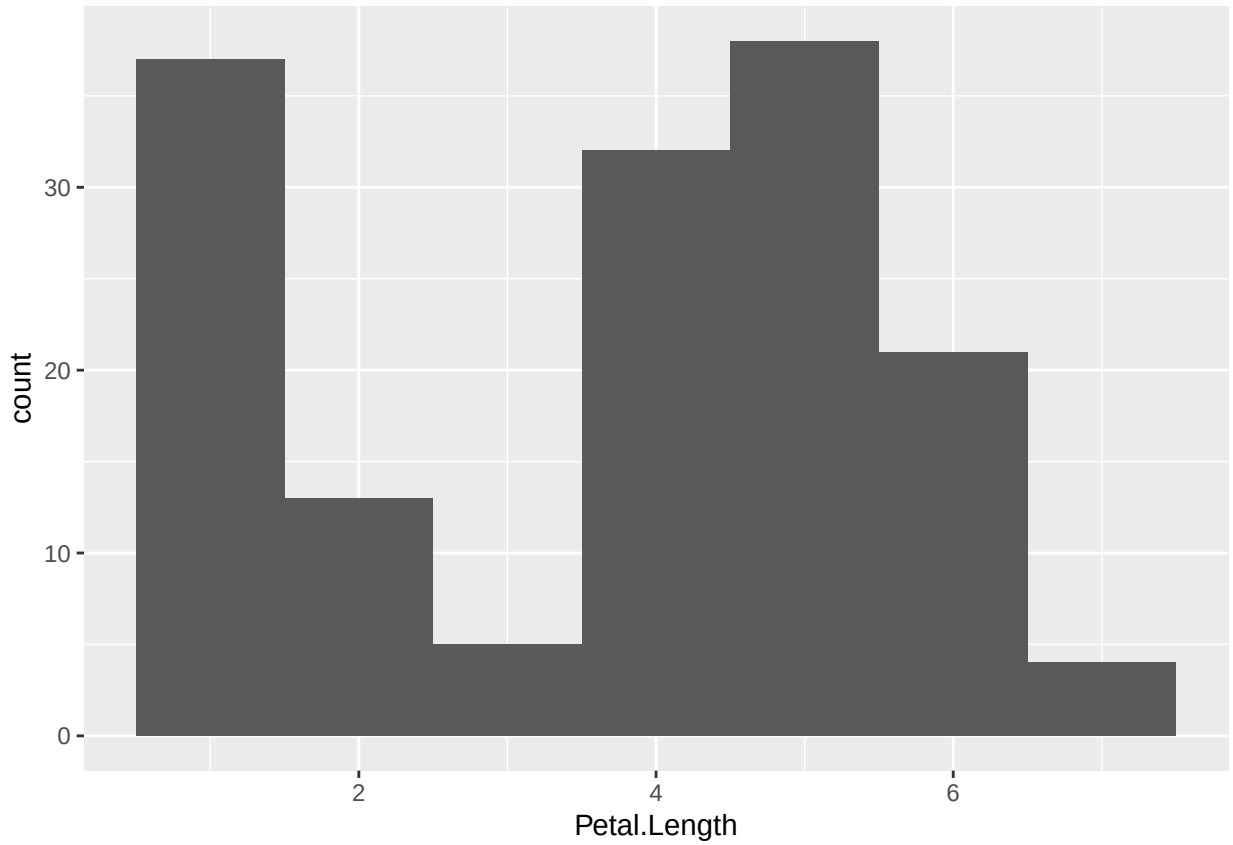



```
petalgg + geom_area(stat="bin") #warning just tells you that default 'bin=30'
```

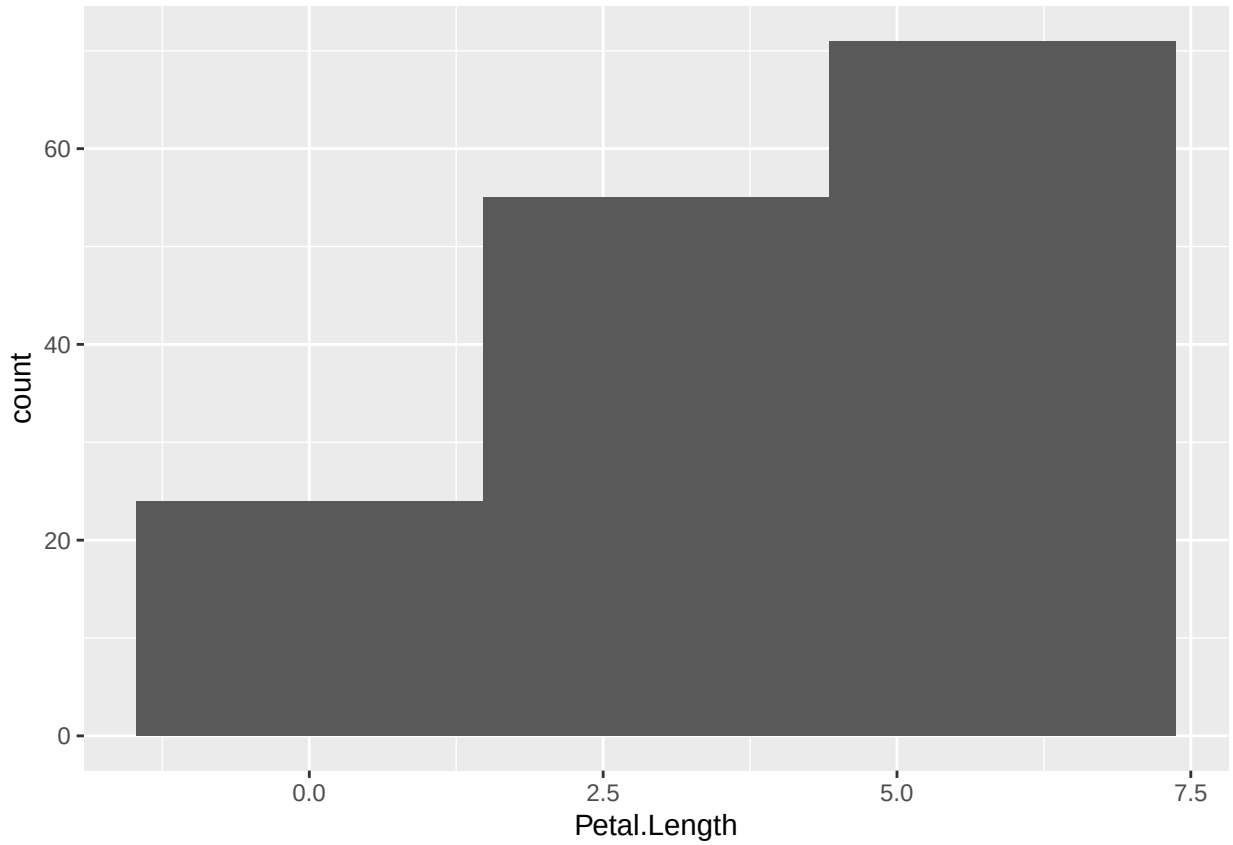
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



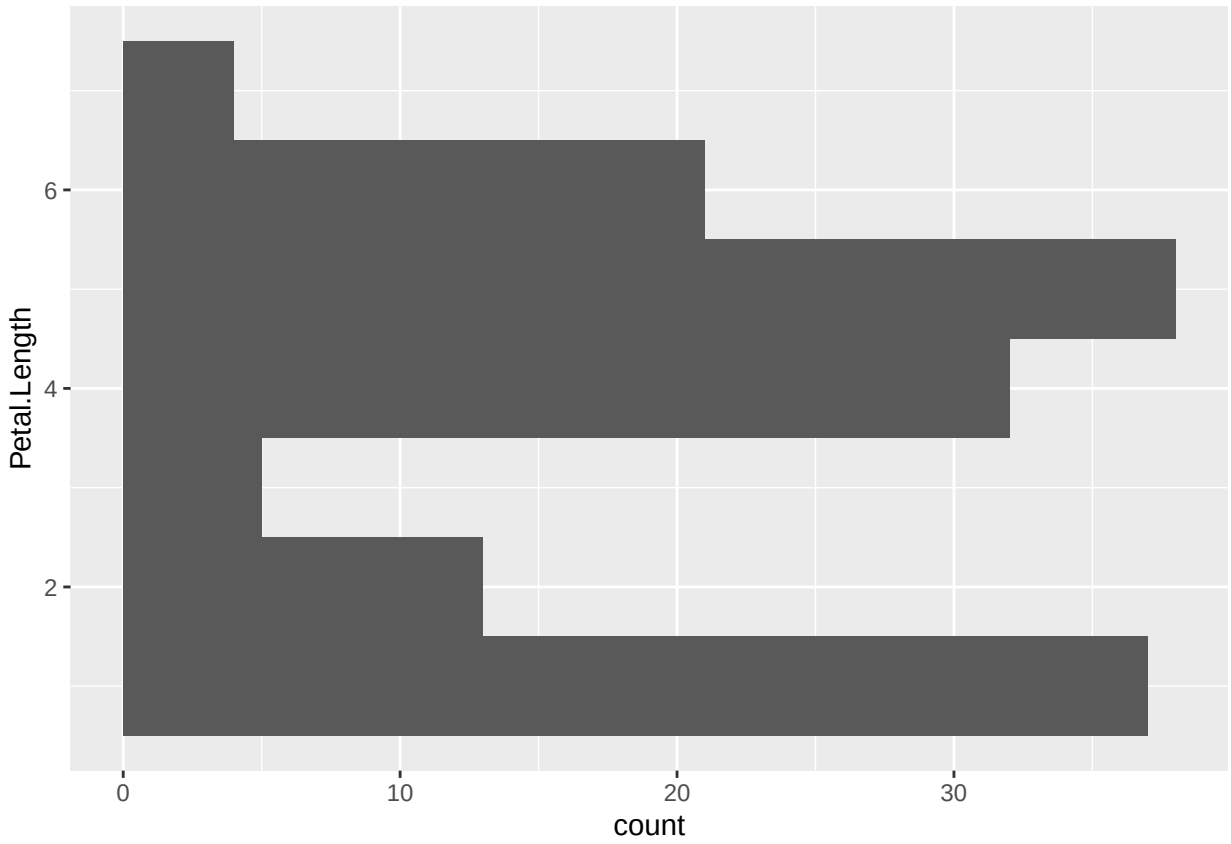

```
# Follow the warning suggestion and explore different bin sizes
petalgg+geom_histogram(binwidth = 1) #sets width of the bins
```



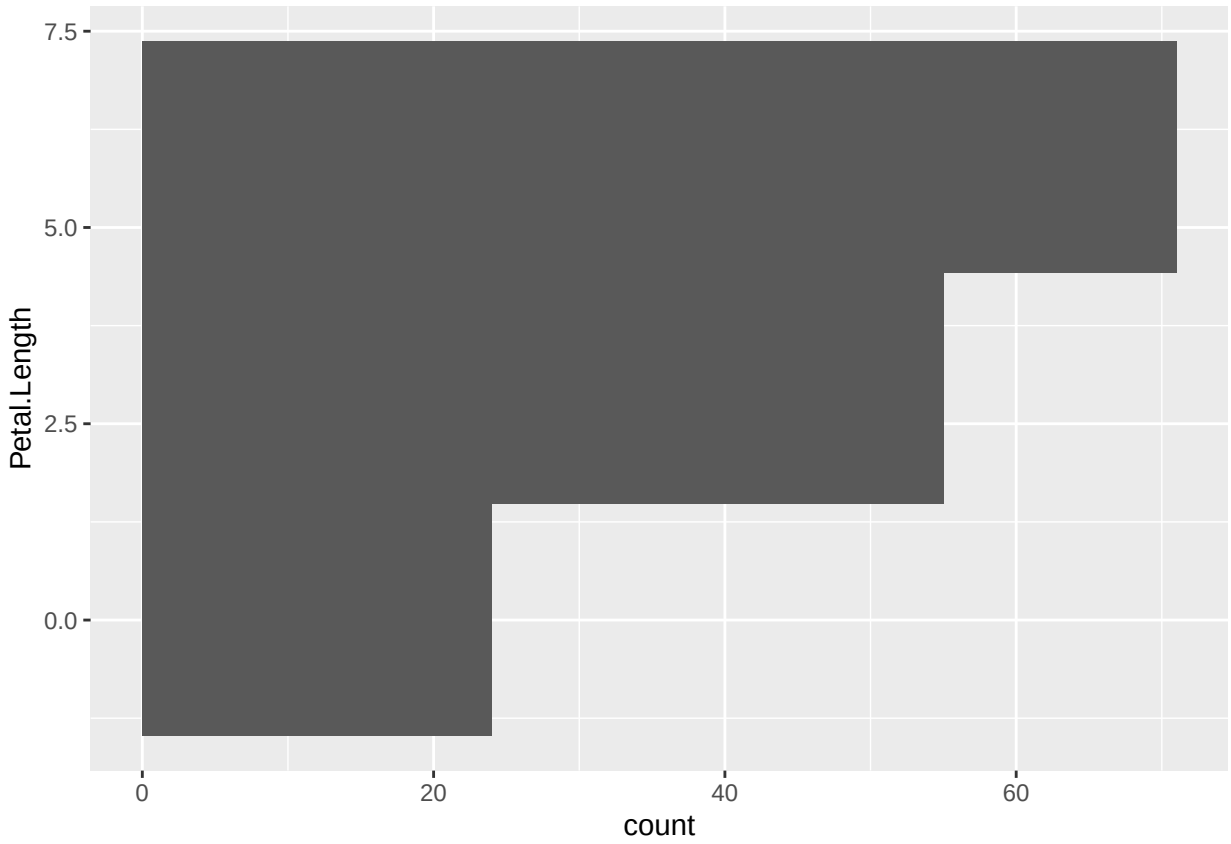

```
petalgg+geom_histogram(bins = 3) #sets number of bins TO THREE. Vindication!
```




```
# Try to flip the axis order of your plot
petalgg+geom_histogram(binwidth = 1)+coord_flip()
```




```
petalgg+geom_histogram(bins = 3)+coord_flip()
```

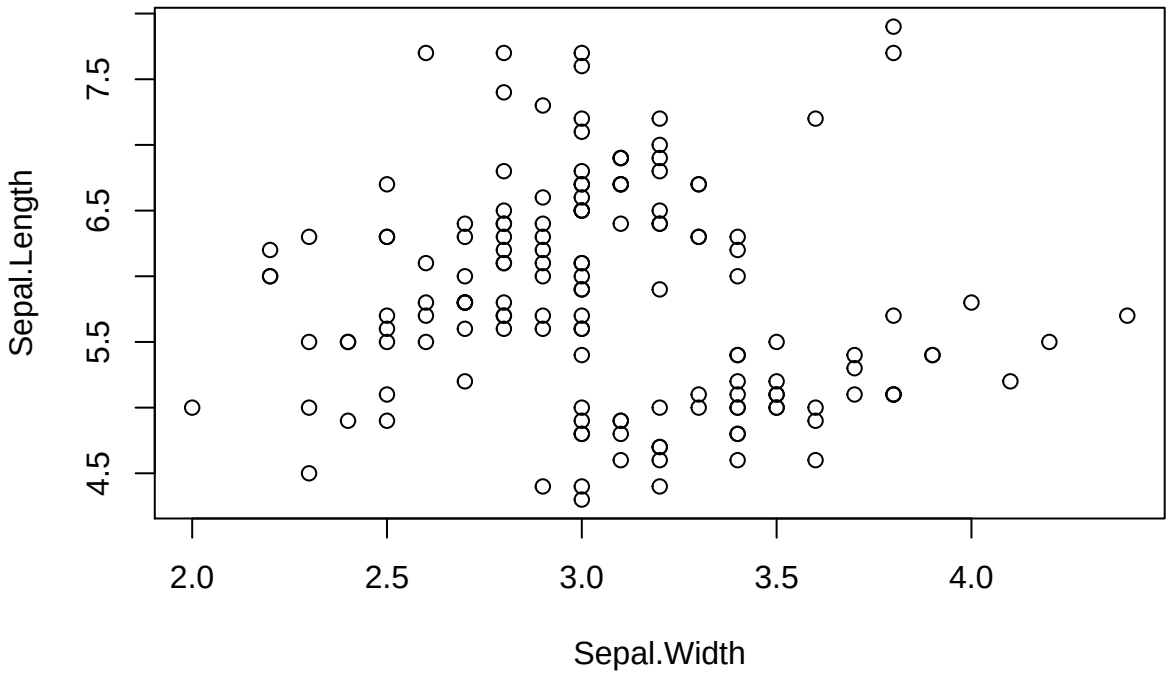



```
# hashtagflipped B)
```

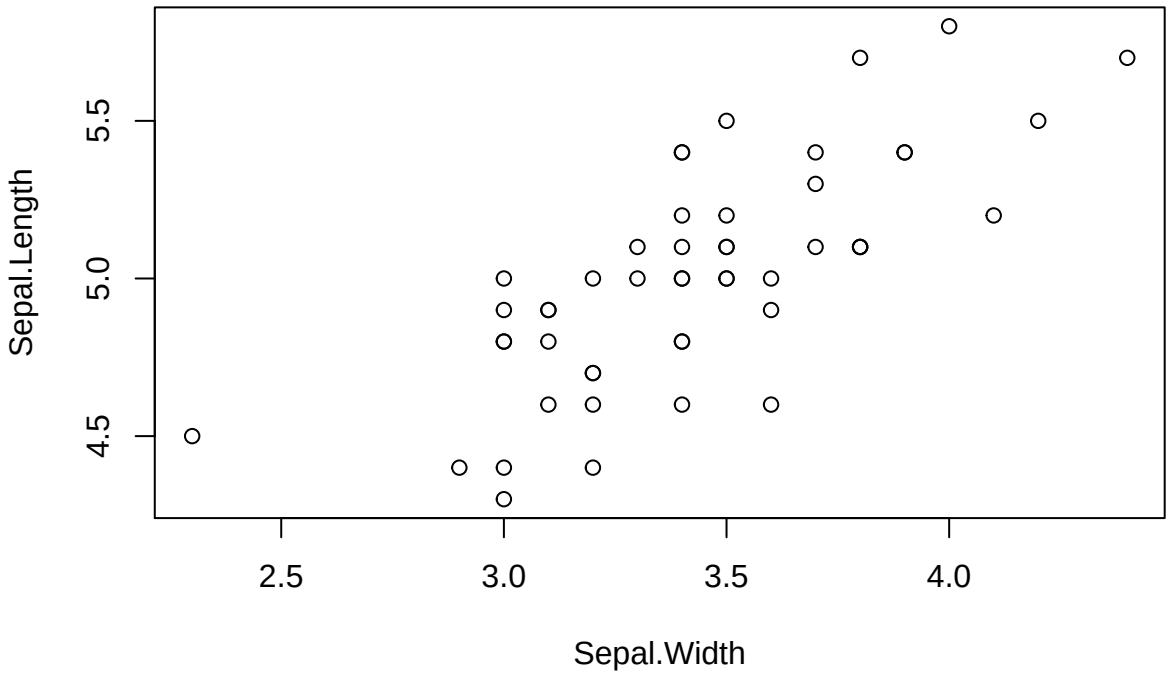
Plot Two Variable - Base

Continuous X, continuous Y

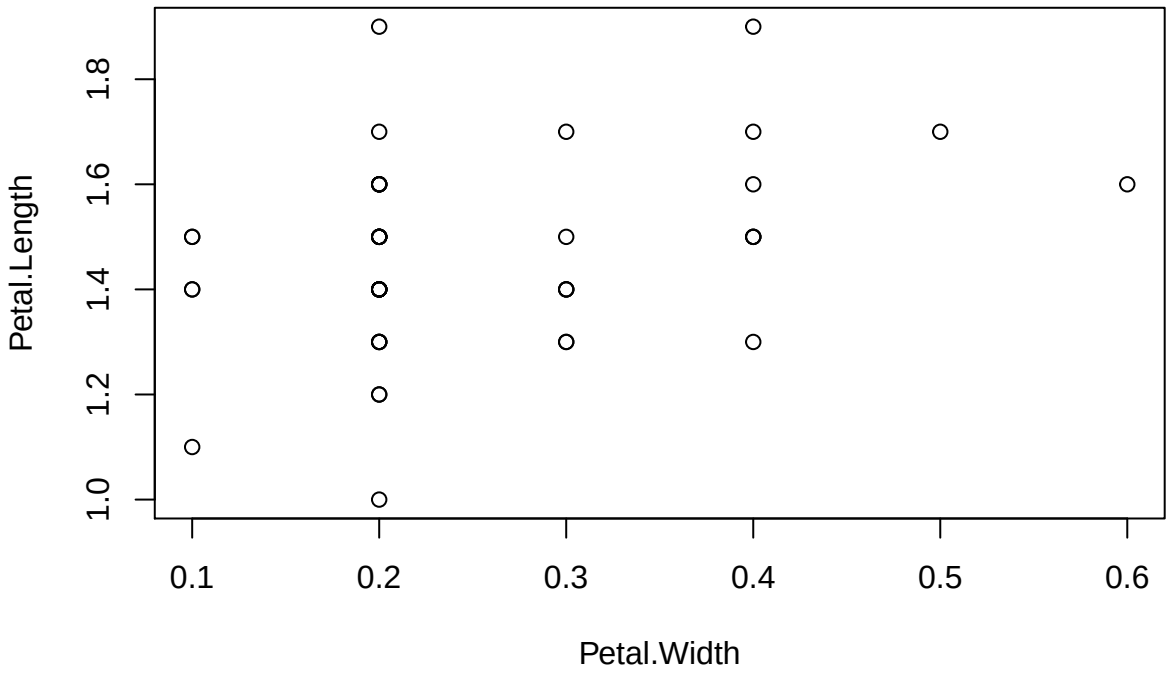
```
# Explore different plots interchanging the 'predictive' and 'response' variables among the four availa
#For predictive variable sepal width, response variable sepal length:
plot(Sepal.Length~Sepal.Width, data=irisdata)
```



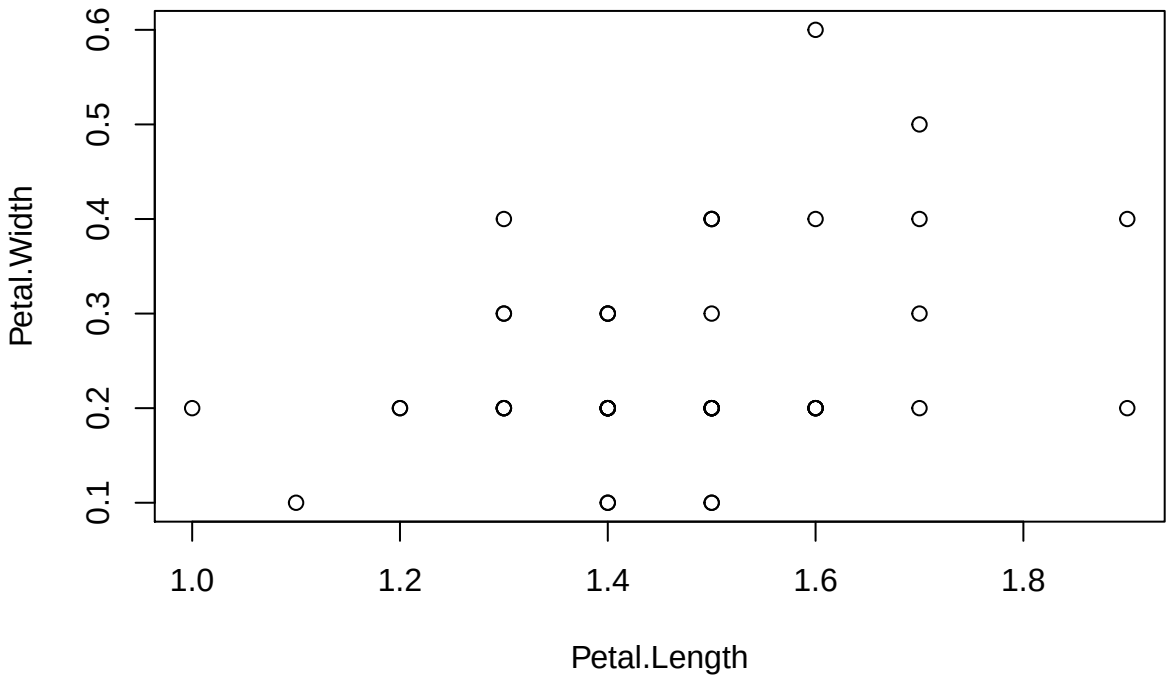

```
#there are three species of flower in this dataset,
#which messes with the ability to have a clear relation between variables. SO:
setosa<-filter(irisdata, Species=="setosa")
plot(Sepal.Length~Sepal.Width, data=setosa) #this one looks like there might be a relationship.
```




```
#Some other attempts:
plot(Petal.Length~Petal.Width, data=setosa)
```

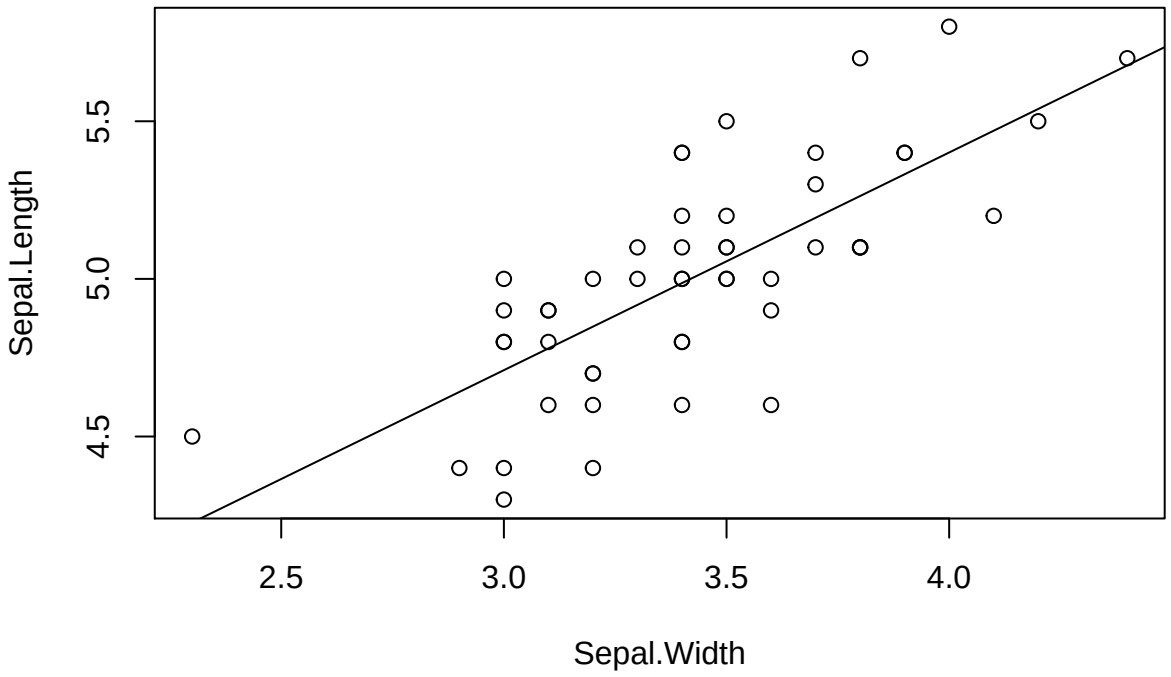



```
plot(Petal.Width~Petal.Length, data=setosa)
```




```
#it looks like the width and length of the sepal has a more defined proportion than the width and lengt

# Now I'll add a line representing linear model fit to the more successful graph:
plot(Sepal.Length~Sepal.Width, data=setosa)
abline(lm(Sepal.Length~Sepal.Width, data=setosa))
```



Discrete X, continuous Y

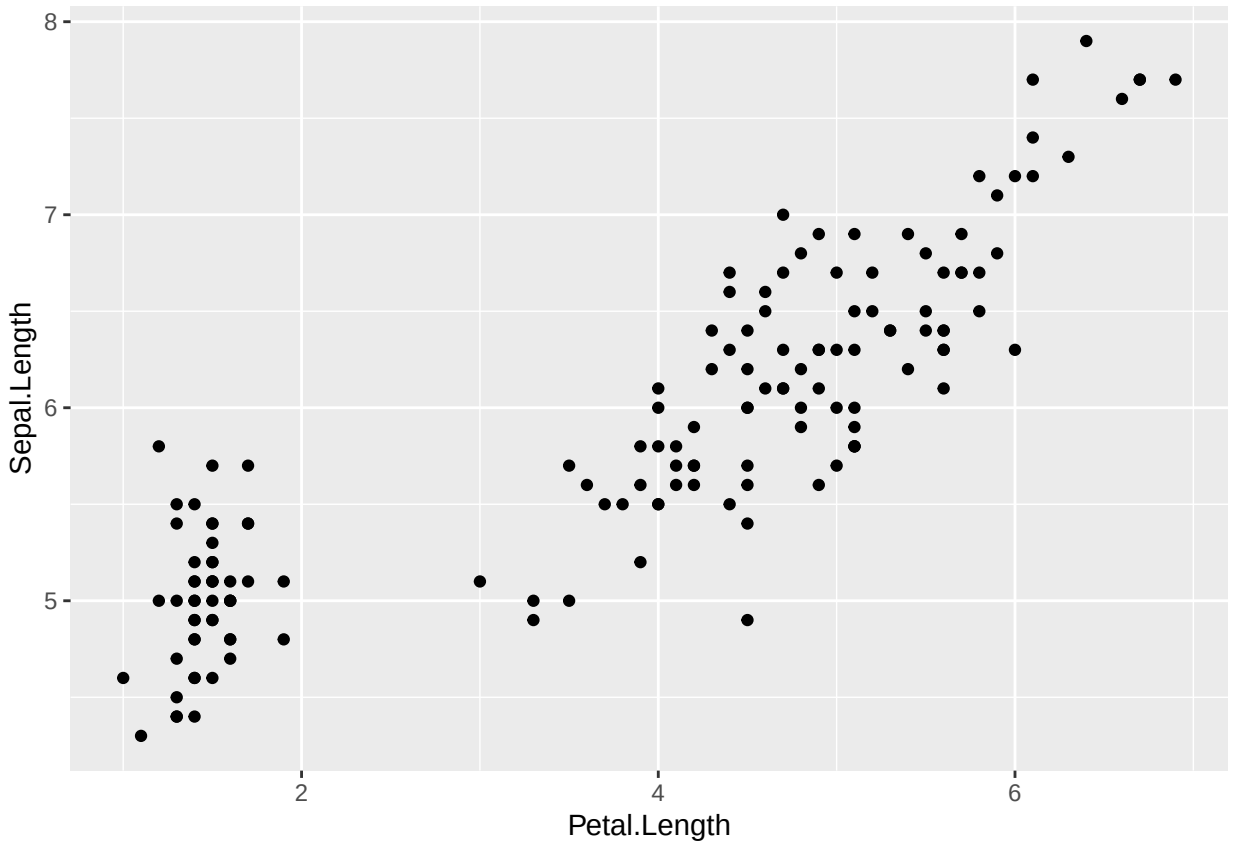
```
# Make a plot that most accurately represent the whole range of a continuous (Y) by species (X)

#IDONOTKNOW
```

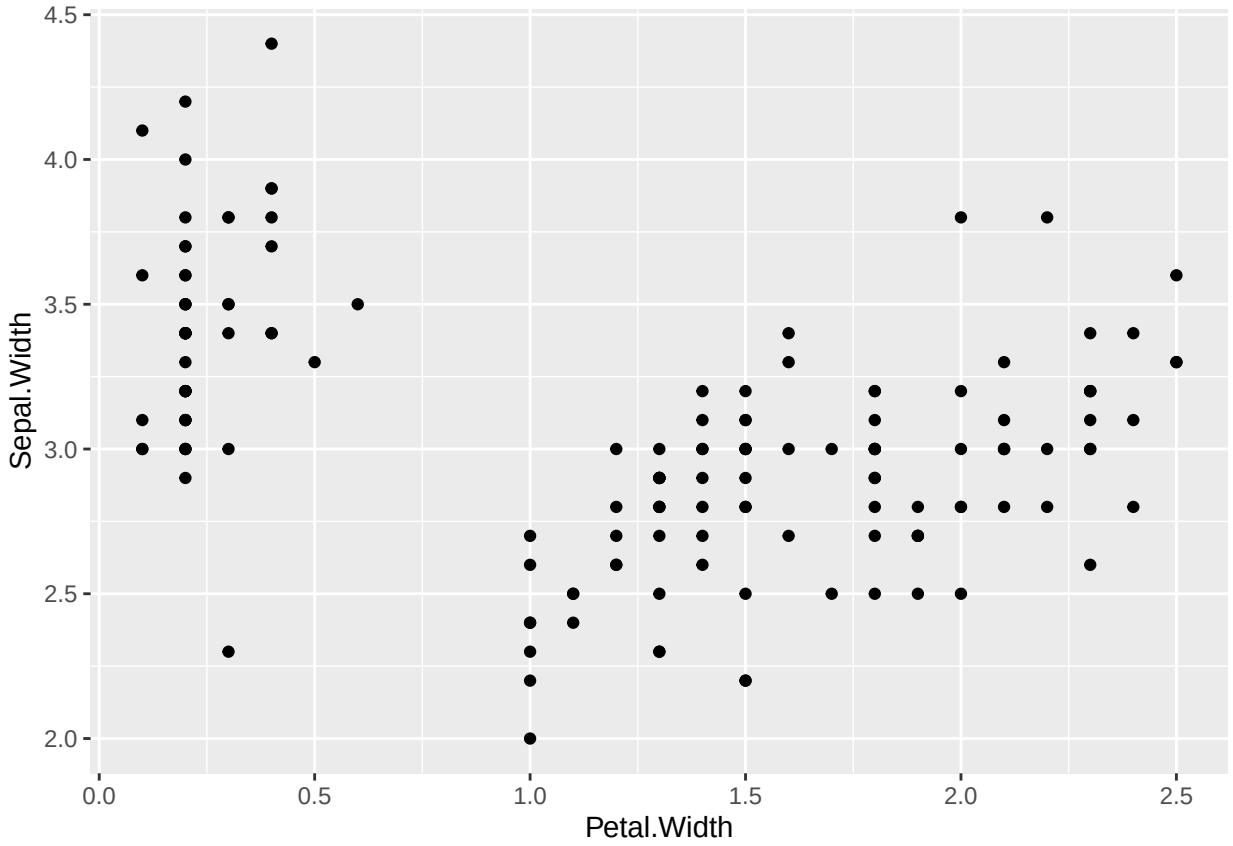
Plot Two Variable - ggplot

Continuous X, continuous Y

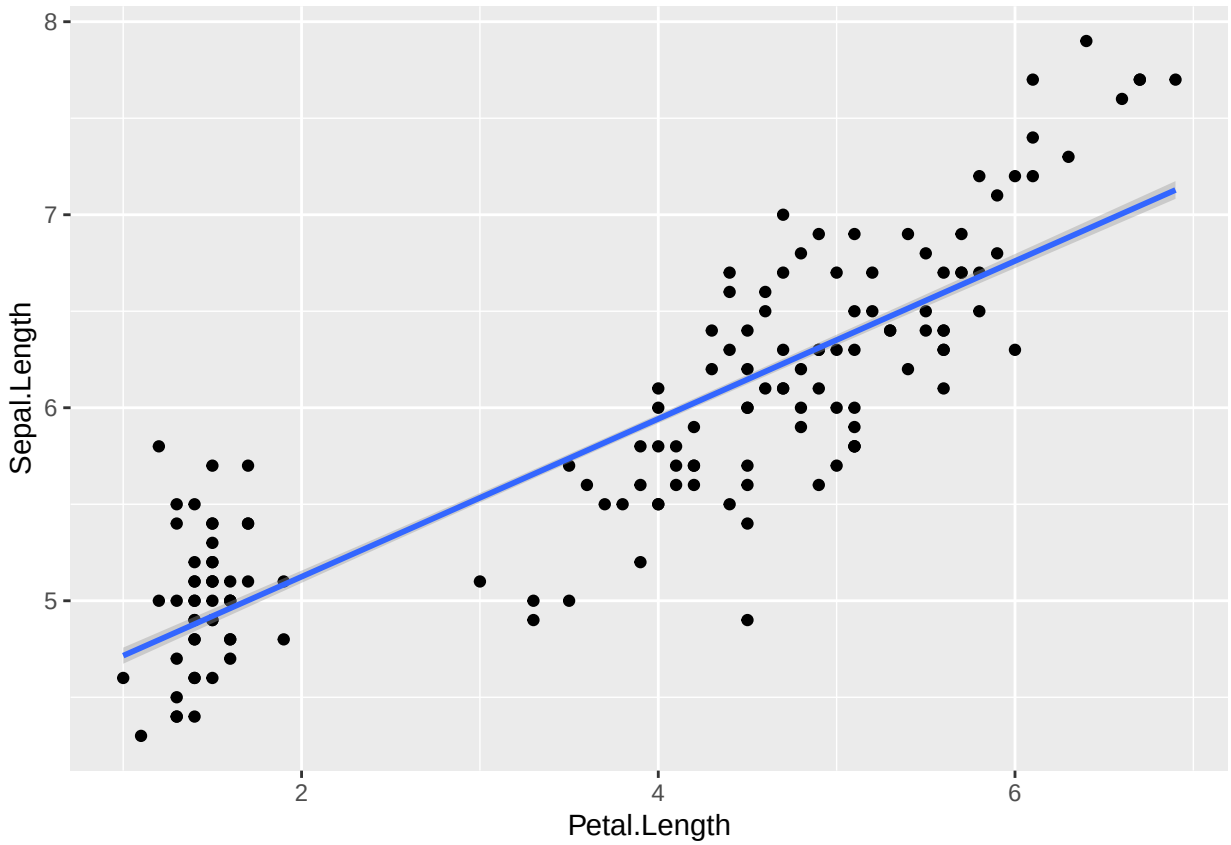
```
# Explore different plots interchanging the 'predictive' and 'response' variables among the four availa
#for this one I will plot petal length and sepal length
flowersize<-ggplot(data=irisdata, aes(x=Petal.Length, y=Sepal.Length))
flowersize+geom_point()
```




```
#now petal width and sepal width
flowerwidth<-ggplot(data=irisdata, aes(x=Petal.Width, y=Sepal.Width))
flowerwidth+geom_point()
```

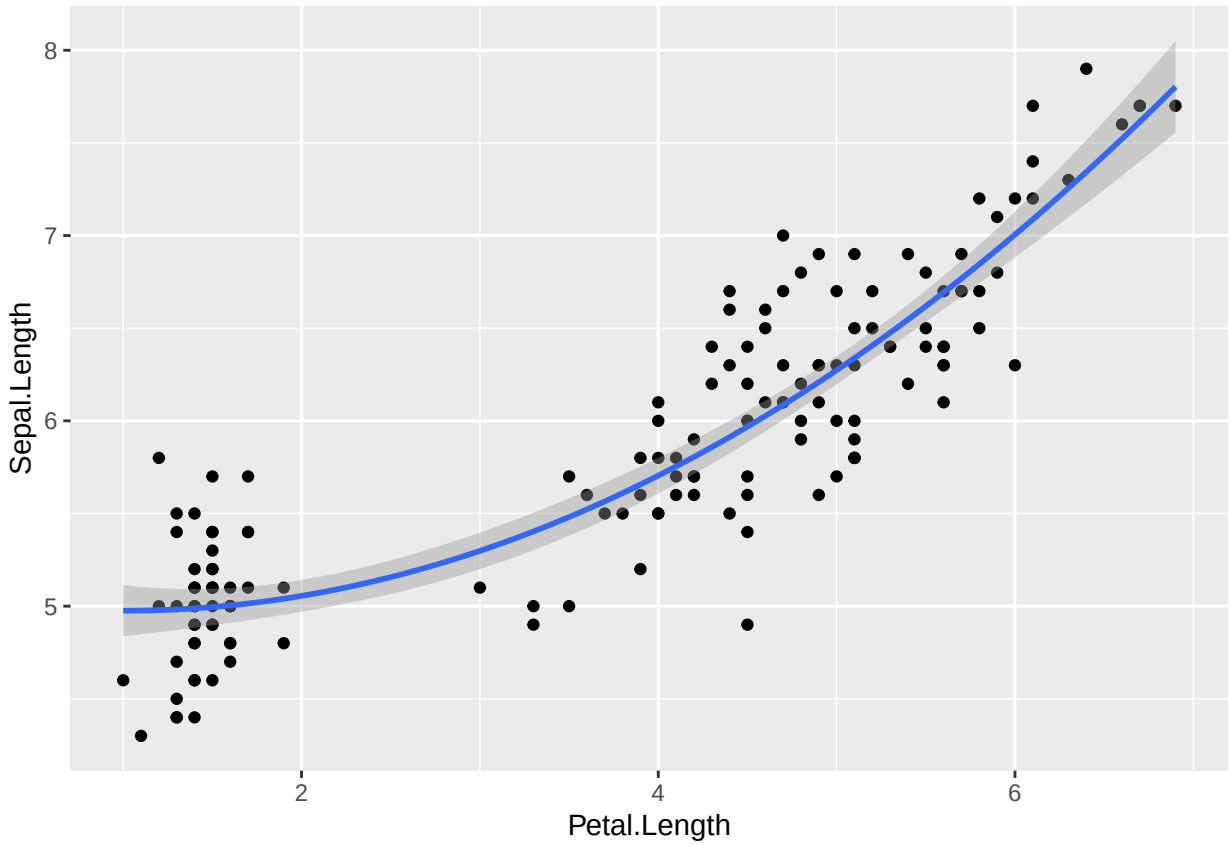



```
#not nearly as neat, it looks like different species might have different relationships
# Try adding lines representing different trend lines to help you explore patterns
# Add a linear model fit to your plots
#I'll do this to my more successful plot
flowersize+geom_point()+geom_smooth(method="lm", level=0.5)
```



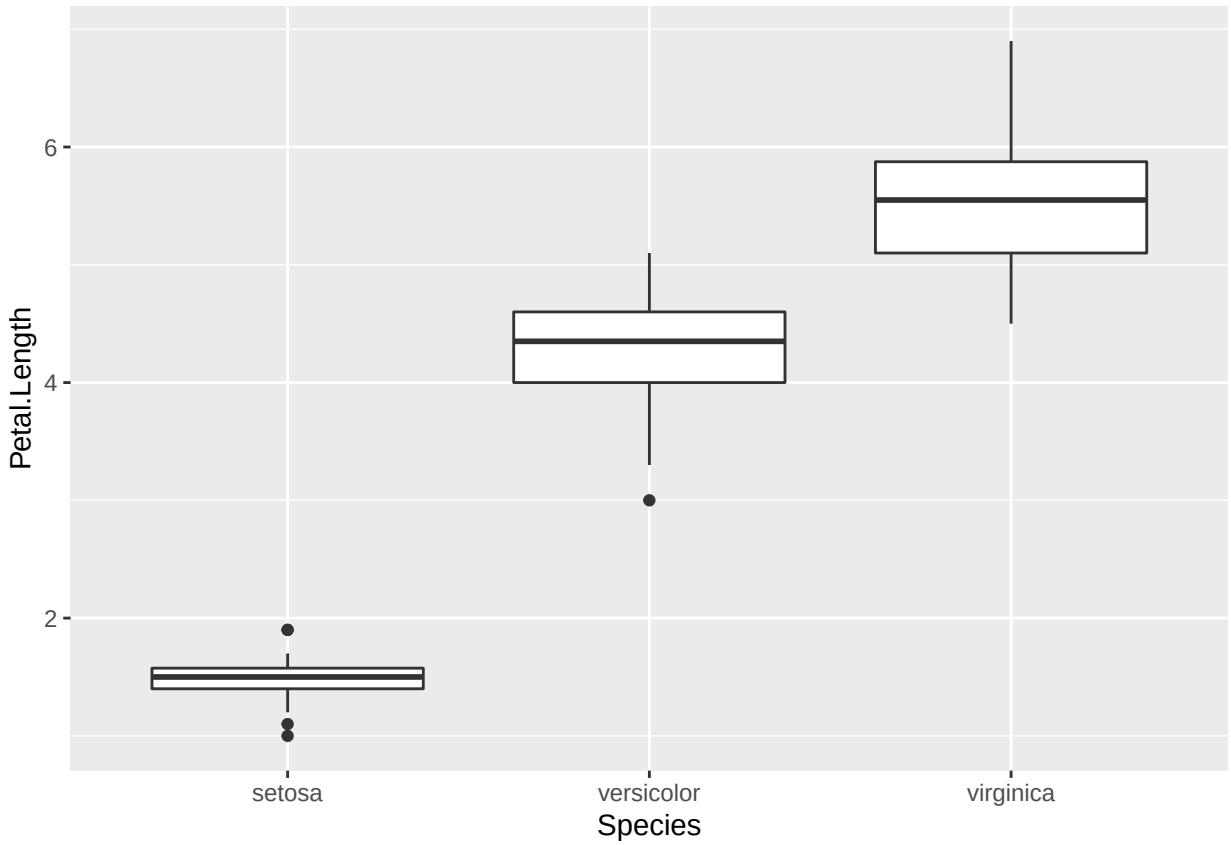

```
# Add a 'Loess smoothing' fit to your plots
 flowersize+geom_point()+geom_smooth(span=100)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

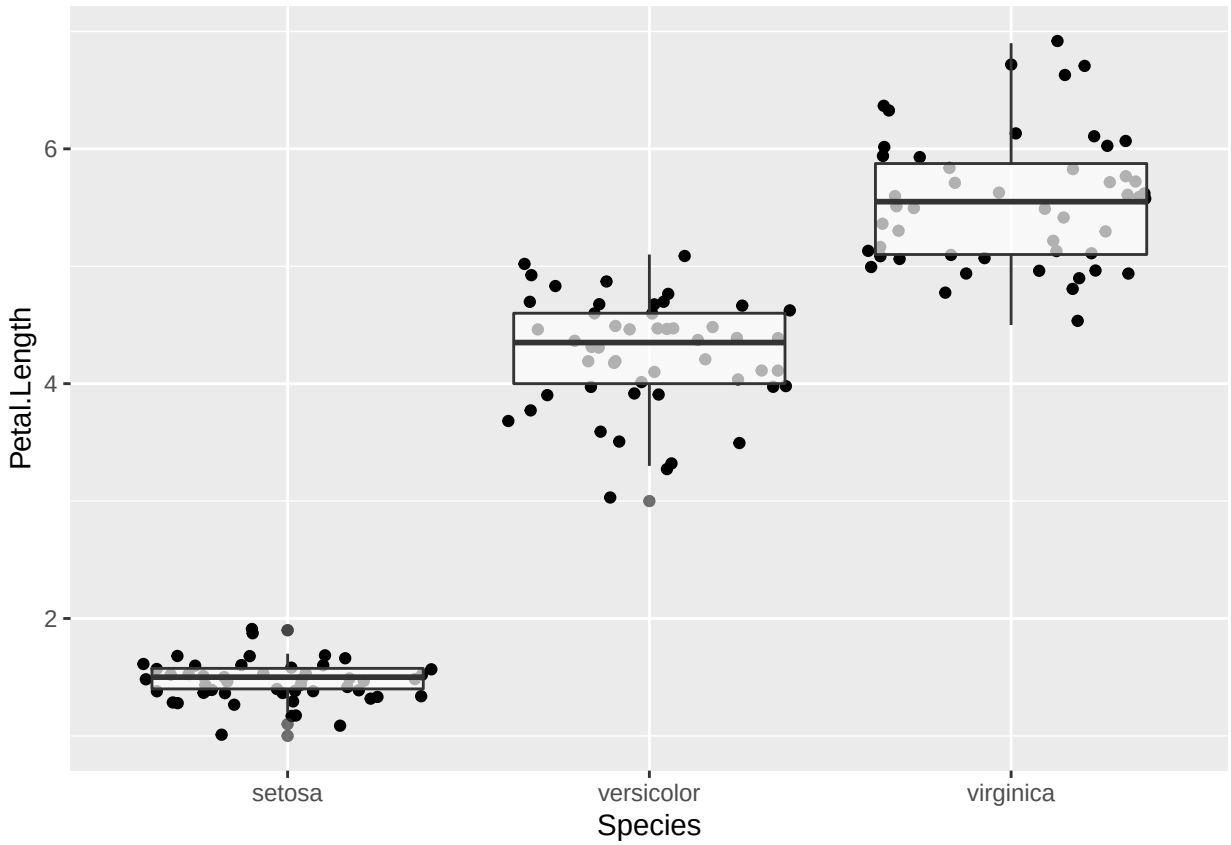


Discrete X, continuous Y

```
# Make a plot that most accurately represent the whole range of a continuous (Y) by species (X)
ggplot(data=irisdata,aes(x=Species,y=Petal.Length))+geom_boxplot()
```




```
# Try adding datapoints to this plot to reveal patterns not easy to see in a whiskerplot
ggplot(data=irisdata,aes(x=Species,y=Petal.Length))+geom_jitter()+geom_boxplot(alpha=0.7)
```




```
#Et voila
```